

A documentation of the

Face Detection and Quick Response Code Scanner System

A partial fulfillment of the requirements for the midterm of first semester in the course Modelling and Simulation

Polytechnic University of the Philippines
Department of Computer and Information Sciences
BSCS 2-3
Group 4

Table of Contents

Table of Contents	2
Member Roles	3
Project Description	3
System Snapshot	4
System Pseudocode and Flowchart	7
Libraries	11
Project Features Source Code (Back-end)	12
Face Detection	12
Face Recognition	14
Addendum	16

Members and Developers

Project Manager and System Developer

Calendario, Mark Kenneth

User Interface and User Experience Designers

Abarre, Jeo

Sinalubong, Sean Matthew

System Analyst

Sinalubong, Sean Matthew

Business Analyst

Cruz, Jan Miles

Technical Designers

Gani, Zaimmon

Yim, Gwyneth Ann Marie

Technical Writers

Brigola, Jaycee

Vedasto, Micole Aaron

Project Description

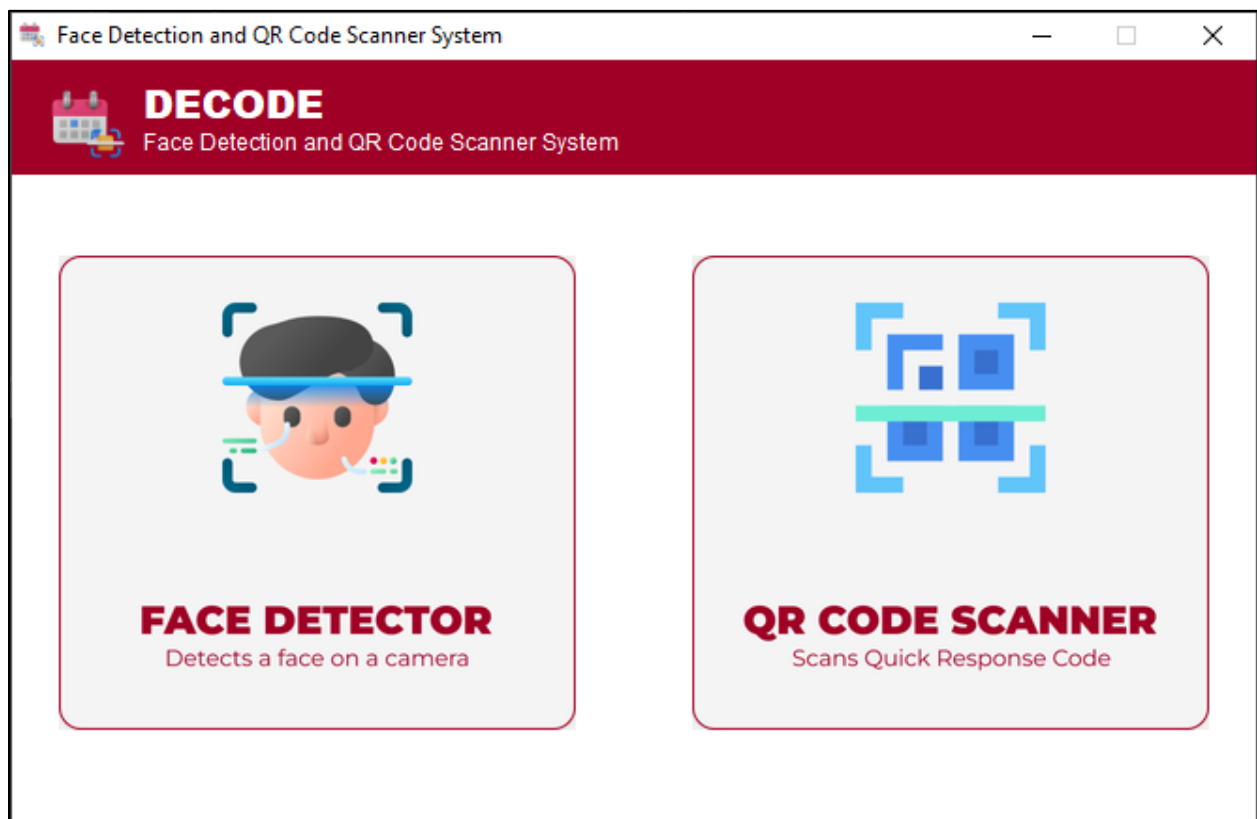
The QR Code Scanner and Face Detection System is a combination of two useful features: quick response code scanning and decoding and frontal face detection. The QR code scanning feature allows users to decode and get information encoded in a machine-readable black and white matrix by simply pointing their device's camera at a QR code, while the face

detection feature uses advanced computer vision algorithms to detect and track faces in real-time. This project is designed to be user-friendly, with a simple and intuitive interface.

System Snapshot

Main Frame of the System

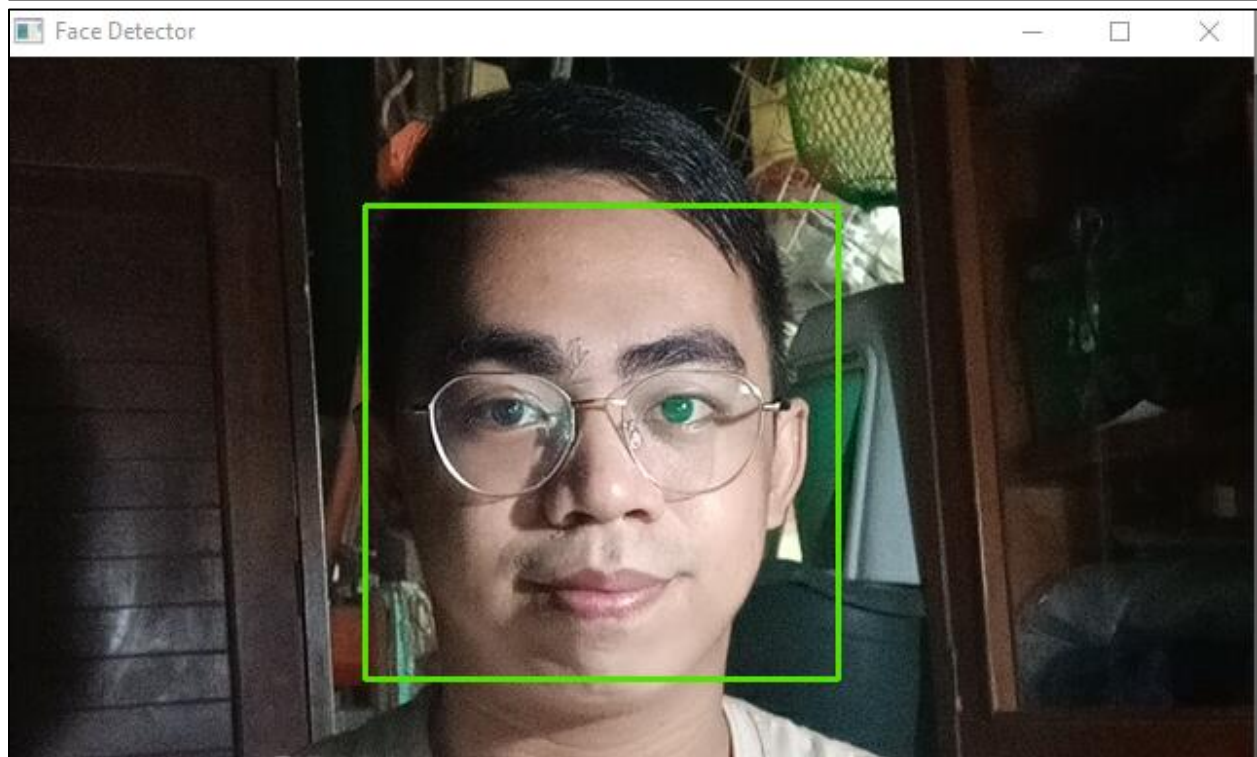
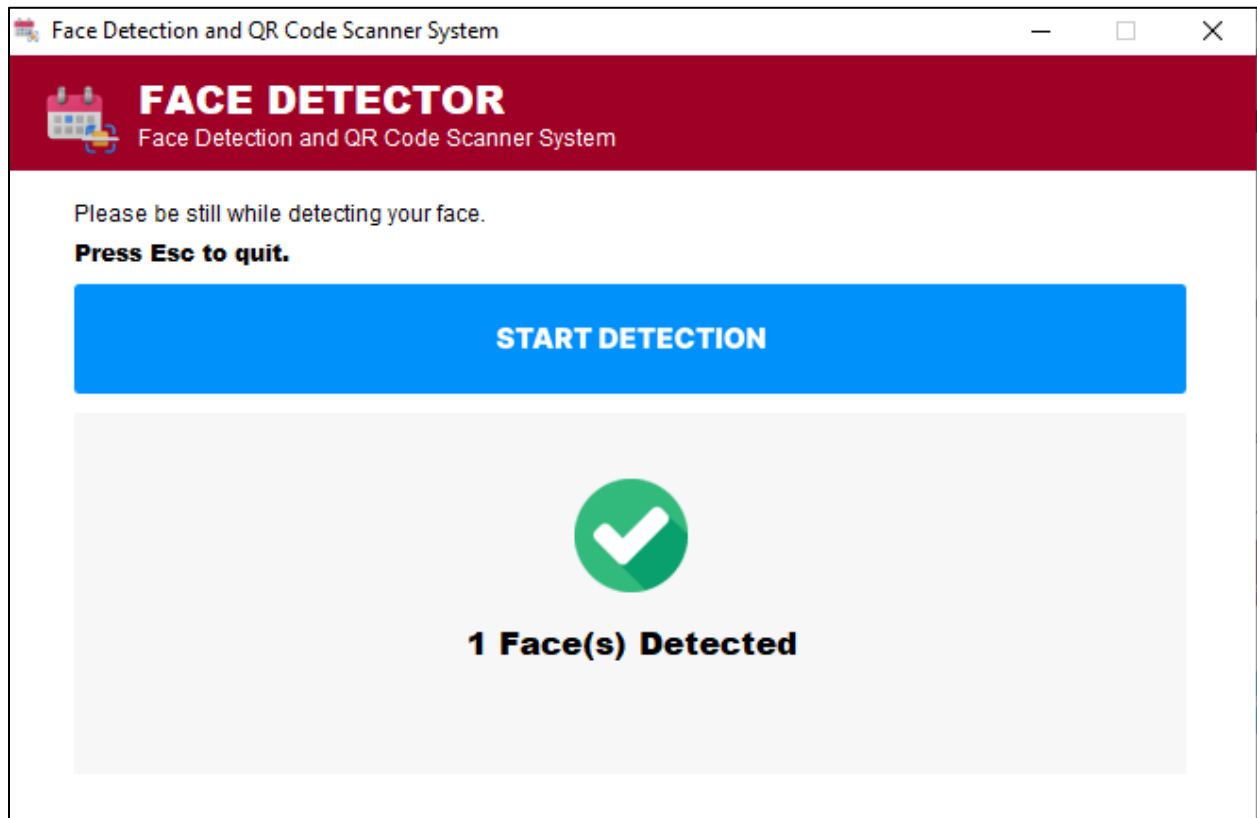
Upon launching the application by clicking on the executable file (exe), the main frame of the application will appear. The root layer of the graphical user interface contains two buttons for the main features of the system: a face detector and a QR code scanner.



Face Detection Data Frame

This Face Detector frame is stackable from the root layer of the GUI. It contains the control for starting the detection and the guides on how to use and exit the frame. This frame is responsible for displaying the real-time state of the camera device, face detection, and faces count

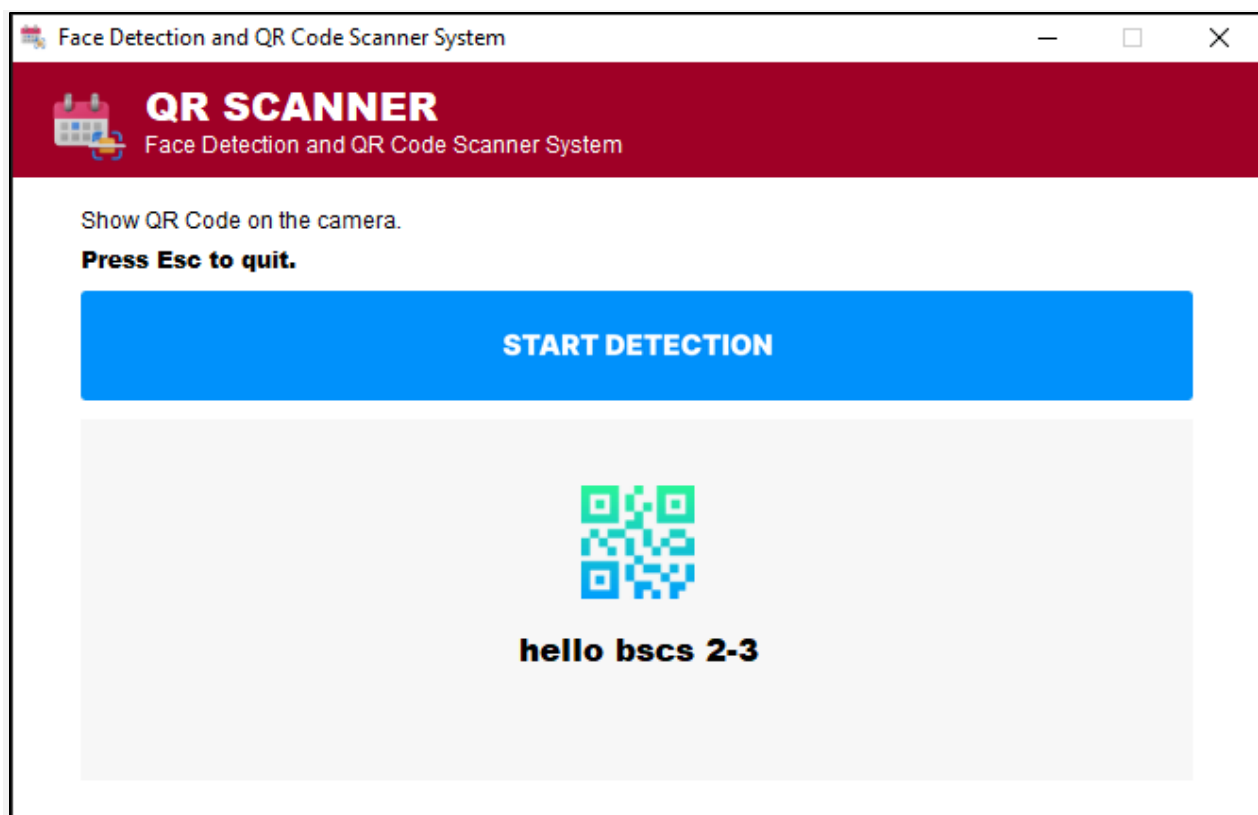
on the frame. The start detection button is placed on this frame for starting up a real-time camera stream wherein the face detection will take place.



QR Code Scanner Data Frame

This QR Code Scanner Data Frame is also stackable from the root layer. Once the QR Code Scanner button was clicked from the main frame, the frame below will popup. This frame consists of real-time data of decoded QR code and other indicators from the camera stream.

The start detection button is placed and packed on this frame for showing up a real-time camera window for the QR Code scanning to take place.





System Pseudocode and Flowchart

This portion of this paper will discuss and show the flowchart or diagram representation of the system and pseudocode or flow of the system Face Detection and Quick Response Code Scanner System in text format. The flowchart and pseudocode of the system is divided into three parts:

1. Main Panel of the System
2. Face Detection Data Frame
3. QR Code Scanner Data Frame

Root Layer of the App

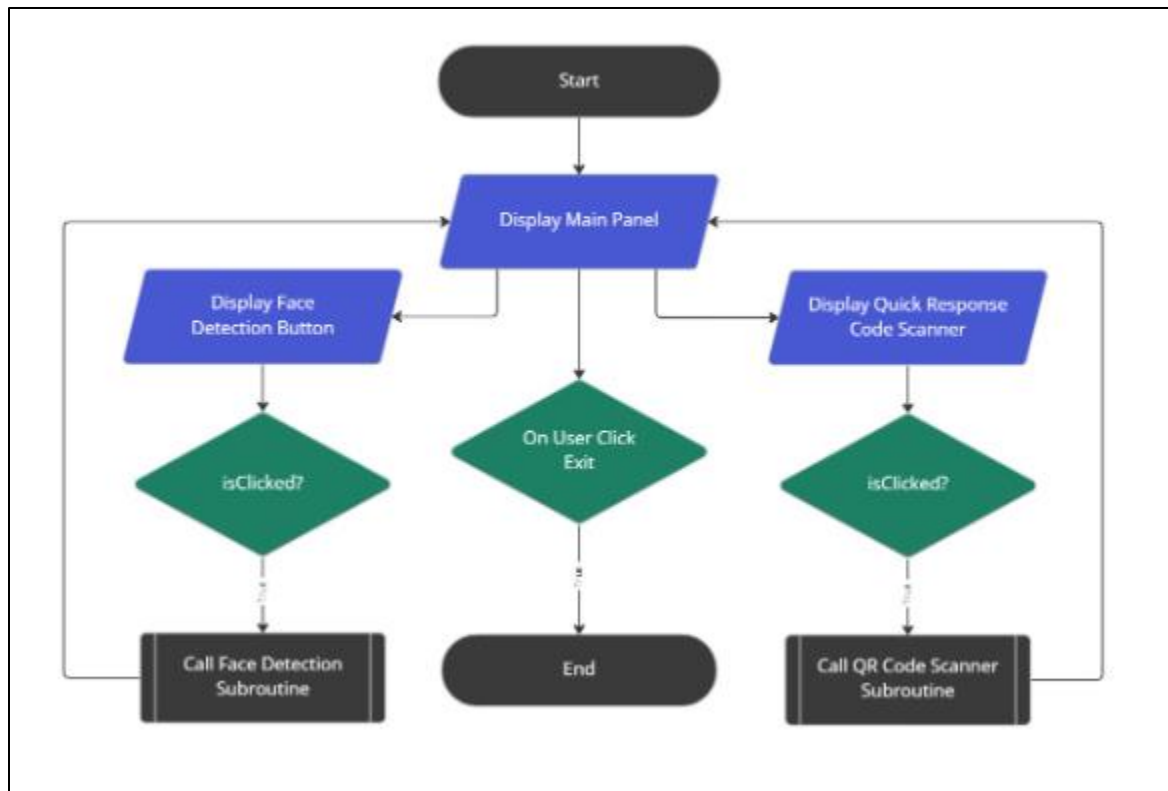
This algorithm shows how the root layer of the application will work. First, it initializes the GUI and display it on the computer screen. Then, it will wait to the user click events.

If user clicked the face detection button, the face detection frame will pop up. After that, the system will check if the user has a camera, if so, it will call the Face Detector subroutine to start the face detection. Otherwise, it will display "you don't have a camera".

However, if user clicked the QR Code Scanner button instead, the QR code scanner frame will appear. The system will check if user has available camera, the system will call the QR Code Scanner subroutine. If the user does not have available camera, the message box with “you don’t have a camera” text will popup.

The application will exit if user click window ‘X’ button.

```
1 ALGORITHM FaceAndQRScanner
2   DISPLAY MAIN PANEL
3   LISTEN to USER EVENTS
4
5   IF USER EVENT == Face Detection Click THEN
6     Open Face Recognition Frame
7
8     IF USER HAS CAMERA THEN
9       Open Camera Live Feed Window
10      Call SubRoutine FaceDetector
11    ELSE
12      DISPLAY “YOU DON’T HAVE A CAMERA”
13
14    Go Back to MAIN PANEL
15
16    IF USER EVENT == QR Code Scanner Click THEN
17      Open QR Code Recognition Frame
18
19      IF USER HAS CAMERA THEN
20        Open Camera Live Feed Window
21        Call SubRoutine QRCodeRecognition
22      ELSE
23        DISPLAY “YOU DON’T HAVE A CAMERA”
24
25    Go Back to MAIN PANEL
26
27  IF USER EVENT == Click Window Close THEN:
28    END
```

Face Detection Data Frame

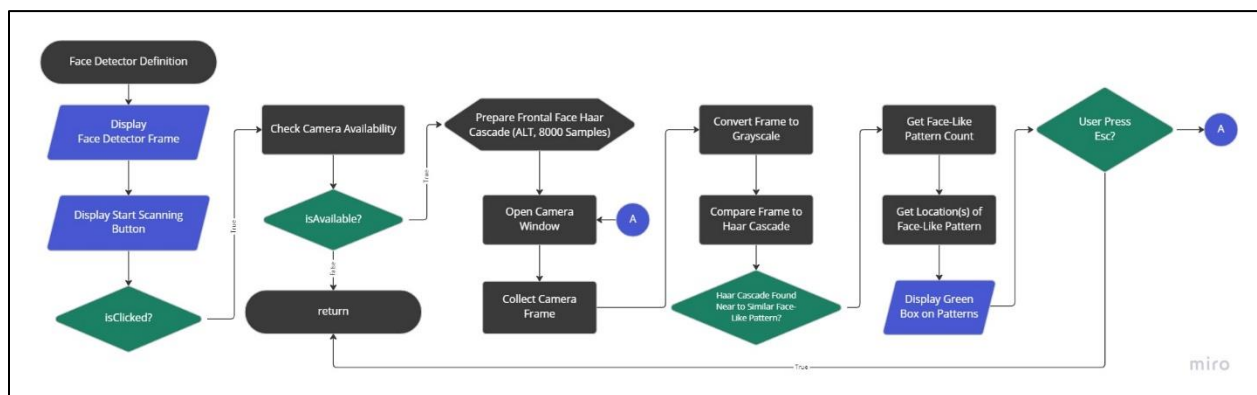
This algorithm indicates the process flow of the face detection data frame. Initially, it reads the frontal face Haar Cascade's XML file. Then, each frame being grabbed from the camera stream will be converted into grayscale for better accuracy. Then, if the face has near to similar pattern on Haar Cascade. After that, the logic will count and get the position of the detected frontal face on the frame. Then, the green box will also appear based on the position of faces. Lastly, it will display the count of faces on the GUI.

If user decided to click the 'Esc' button, the camera and current frame will exit. Then the user will be navigated to the main panel of the system again.

```

1 SUBROUTINE FaceDetector
2   Read Frontal Face Haar Cascades XML File
3   Convert each frame to grayscale
4
5   IF face has near to similar pattern on Haar Cascade THEN:
6     Get count of frontal face on frame
7     Get position of frontal face/s
8     Display green box on frontal face/s
9     Display count of faces on GUI
10
11   IF USER EVENT == Pressed 'Esc' THEN:
12     Close camera
13
14 RETURN

```



QR Code Scanner Data Frame

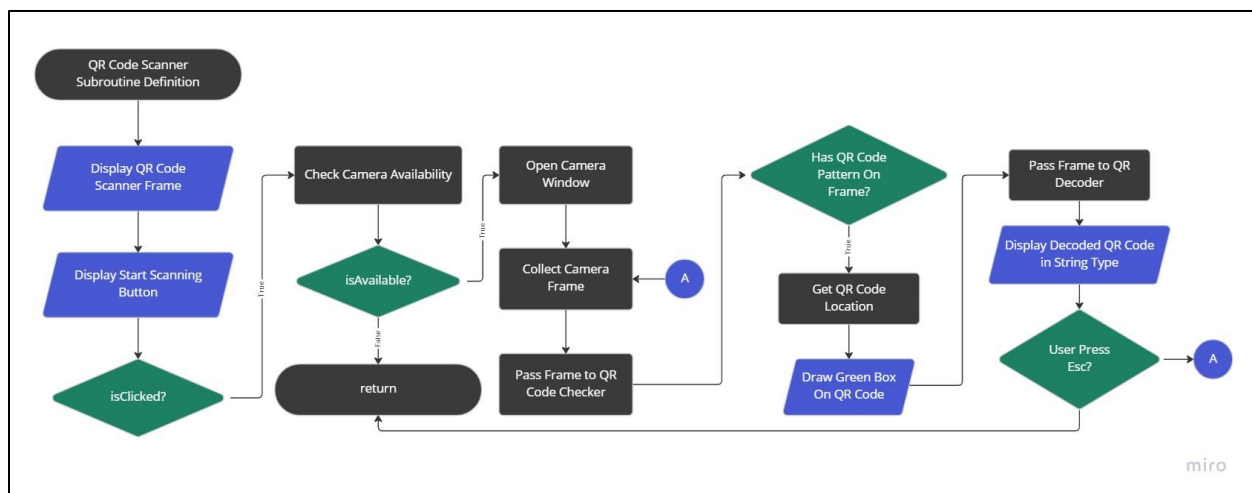
This algorithm shows the flow of the Quick Response scanner data frame. The logic behind this feature is basically grabs each frame from the camera stream and it is being passed to the open-source computer vision (open-cv) library and scan the frame if there are any QR code. If there is QR code, then the system will get the position of the QR code and display a box on top of it. Moreover, the open-cv will decode the QR code and get the string type result. Lastly, the system will display the result right beside the box.

If user pressed 'Esc', then it will close the camera and will go back to the main frame.

```

1 SUBROUTINE QRCodeRecognition
2   Get each frame from camera stream
3   Pass each frame to opencv
4   Scan each frame for QR Code
5
6   IF QR Code Detected THEN:
7     Display box on top of QR Code
8     Decode QR Code
9     Get QR Code Result
10    Display QR Code result beside the box
11
12   IF USER EVENT == Pressed "Esc" THEN:
13     Close camera
14
15 RETURN

```



Libraries

1. **OpenCV** - a free and open-source computer vision library that provides tools for image and video processing.

1.1 **Frontal Face Haarcascade V2** - a model of nine thousand face samples compiled in one XML file for object detection and face recognition functions.

2. TKinter - is a Python library for building graphical user interfaces (GUIs) with the Tk toolkit. It provides tools for creating buttons, labels, text boxes, and menus, as well as support for layout management and event handling.

Project Features Source Code (Back-end)

This portion discusses the back-end source code of the Face Detection and QR Code Scanner feature. The developers ensures that the source code is properly commented and the tried to manage to make it understandable to anyone.

Face Detection Feature

Initialization of frontal face haarcascade second alternative

```
154 def display_face_detection_feed(self):
155
156     # Load the Haar cascade file for detecting faces
157     # Haarcascade is a detection model for a particular object.
158     # In this function, frontalface_alt2 is being used.
159     # This haarcascade consists of 9,563 face model.
160
161     face_cascade = cv2.CascadeClassifier(os.path.join(OUTPUT_PATH, 'haarcascades',
162     'haarcascade_frontalface_alt2.xml'))
163
164     # Initialize the video capture object
165     available_camera_index = get_available_camera()
```

Definition of get_available_camera()

```
3 def get_available_camera():
4
5     # Allow loop to test camera devices from -50 to 100
6     for camera_index in range(-50, 101):
7         camera = cv2.VideoCapture(camera_index)
8         is_cam_capturing, _ = camera.read()
9         camera.release()
10
11     # If the camera index is available then return the index
12     if is_cam_capturing:
13         return camera_index
14
15     # If there are no camera available, return None
16     return None
```

On `get_available_camera()` return None

```
166 # If no available camera device
167 if available_camera_index == None:
168     messagebox.showerror("Camera is not detected", "You do not have available camera.
169     Please make sure your camera is turned on and connected.")
170     self.top.destroy()
171     self.root.deiconify()
172     return
```

Face Detection under the hood process

```
173 # Open a camera
174 camera = cv2.VideoCapture(available_camera_index)
175 is_camera_capturing, frame = camera.read()
176
177 # While camera is getting frame
178 while is_camera_capturing:
179
180     # Continuously grab the frame from the video capture object
181     is_camera_capturing, frame = camera.read()
182
183     # Convert the frame to grayscale for better detection
184     gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
185
186     # Look for near to similar faces in the frame using the loaded cascade file
187     faces = face_cascade.detectMultiScale(gray, 1.1, 4)
188
189     # Draw a rectangle around the faces
190     for (x, y, w, h) in faces:
191         frame = cv2.rectangle(frame, (x, y), (x+w, y+h), colors.get(
192             'face_detector_box'), 2)
```

Displaying frames on the camera stream

```
193 cv2.imshow('Face Detector', frame)
```

On user termination

```
195 # Terminate if user pressed 'Esc' key
196 if cv2.waitKey(1) == 27:
197     break
```

```
213 # Close the camera
214 camera.release()
215 cv2.destroyAllWindows()
```

Applying UI modification on the frame

```
199     # If there are faces detected,
200     # Then show a check indicator image
201     if len(faces) != 0:
202         self.canvas.itemconfig(self.face_detection_indicator_image, image=self.
            image_check)
203
204     # Otherwise display a warning indicator image
205     else:
206         self.canvas.itemconfig(self.face_detection_indicator_image, image=self.
            image_warn)
207
208     # Display count of faces detected on the canvas
209     self.canvas.itemconfig(self.indicator_text, text=f"{len(faces)} Face(s) Detected")
210     # Take effect the canvas content modification
211     self.top.update()
```

QR Code Scanner Feature

Open a camera stream if user has available camera device

```
191     def start_qr_scan(self):
192
193         # This function starts a QR Code Scanning if button was clicked
194
195         # Get the available camera
196         available_camera_index = get_available_camera()
197
198         # If there are not camera devices found
199         if available_camera_index == None:
200             messagebox.showerror("Camera is not detected", "You do not have available camera.
                Please make sure your camera is turned on and connected.")
201             self.top.destroy()
202             self.root.deiconify()
203             return
```

Grab each frame from the camera

```
205     # If there are camera
206     # Open the camera and grab first frame
207     camera = cv2.VideoCapture(available_camera_index)
208     is_device_capturing, frame = camera.read()
```

Initialize the OpenCV QR Code Detector

```
209     detector = cv2.QRCodeDetector()
```

On camera continuous capture

```
211     # If first frame was successfully grabbed
212     while is_device_capturing:
213
214         # Continuously grab camera frame
215         is_device_capturing, frame = camera.read()
216
217         # Stop scanning if user pressed 'Esc'
218         if cv2.waitKey(1) == 27:
219             break
220
221         # Read and decode QR Code on the current frame
222         frame = self.read_qrcodes(detector, frame)
```

Definition of user-defined `read_qrcodes` function

- * It decodes frame and get the data text and box position
- * Send decoded text to GUI frame and update the GUI
- * If the frame detects no QR code, return the plain frame
- * Else, get the position and draw a rectangle and display text on the screen

```
158 def read_qrcodes(self, detector, frame):
159     # This function read qr code displayed on the given frame
160     # It uses the Open-CV's QRCodeDetector (@params: detector)
161     # and its method detectAndDecode
162
163     # Decodes and get position of the QR Code displayed on the screen
164     qr_data_text, bbox, _ = detector.detectAndDecode(frame)
165
166     # Display a decoded text on the frame
167     self.canvas.itemconfig(self.result_text, text=qr_data_text if qr_data_text else "No
QR Code Detected" )
168     # Take effect
169     self.canvas.update()
170
171     # If there's no qr code data being decoded,
172     # Immediately return the frame
173     if not qr_data_text:
174         return frame
175
176     # Or else, get the position of the QR code and parse it from float into int
177     point_1 = [int(point_1_floats) for point_1_floats in bbox[0][0]]
178     point_2 = [int(point_2_floats) for point_2_floats in bbox[0][2]]
179
180     # Draw a rectangle on the QR code position
181     frame = cv2.rectangle(frame, point_1, point_2, colors.get('qr_code_scan'), 2)
182
183     # Initialize Font
184     font = cv2.FONT_HERSHEY_COMPLEX
185
186     # Call draw_text function to draw a text
187     self.draw_text(frame, qr_data_text, font, (point_1[0], point_1[1] - 38), 1, 2,
(255, 255, 255), colors.get('qr_code_scan'))
188
189     return frame
```

Displaying the returned frame from `read_qrcodes` function on the video stream

```
224     # Display the frame
225     cv2.imshow('Real Time - QR Code Scanner', frame)
```

On user pressed 'Esc' (refer to line 218), stop capturing and destroy windows

```
227     camera.release()
228     cv2.destroyAllWindows()
```

** END OF THE DOCUMENTATION **

Addendum

Decode Team as Developer

The source-code of this project was developed by the Group Four (4) of the sophomore students' section three (3) of the Bachelor of Science in Computer Science in Polytechnic University of the Philippines. This project is designed, coded, and developed duly for the academic purposes only. The team "Decode" managed to choose open-source assets and libraries in accordance with the best of our knowledge. Any critiques and suggestions are open for the public.

Source Code released to the Public

The team "Decode" allows anyone to clone and download the source-code of this project under certain conditions:

1. Anyone can modify, edit, and improve the content of the system.
2. We, "Decode" team, are not accountable to any damages (if this system can) to any hardware or software.
3. No one can sell and use this project for a commercial use.

Source Code Link: <https://github.com/markcalendario/face-detector-and-qr-code-scanner>